Train Arrival Detection and Railway Track Breakage Detection using Wireless Sensor Networks

Ansari, Sarmad Ahmed
Chair of Communication Networks
Technical University of Munich
Munich, Germany
ga83fal@tum.de

Khan, Muhammad Danish
Chair of Communication Networks
Technical University of Munich
Munich, Germany
ga53sab@tum.de

***Abstract*— Railway transport system is considered one of the key factors in any country's social and economic progress. Special attention is required for the maintenance of railway tracks as it can lead to high financial damage and loss of human lives. Several methods have been used to detect the damage in railway tracks employing wired communication. One problem with the wired communication is the difficulty to locate the fault. A simple and easy to implement method for fault detection and localization based on wireless sensor networks is discussed in this report. The health of the railway track is monitored using wireless nodes and vibration sensors. A flexible and energy aware routing algorithm is developed to route the packet from the point of fault to the main station. This system will help in making timely decisions based on the health of the railway tracks to avoid any unfortunate event.**

Keywords—Wireless Sensor Networks, fault detection, vibration sensor, routing

I. Introduction

Railways are one of the largest transport infrastructures in any country and provide an efficient form of transport because of their capacity, speed, and coverage [1]. The high capacity of railways, however, implies a high risk of human lives. Railways are prone to many faults; apart from the carriage itself, any fault in the railway track may also lead to an accident. The railway tracks are subjected to severe contact stresses at the wheel–railhead interface which may lead to broken tracks [2]. For safe operation, a proactive approach is needed i.e. along with the periodic maintenance of the railway tracks, a continuous monitoring of the health of the track is also important.

The length of the railway tracks, however, poses a problem for the continuous monitoring. Since the length of the railway tracks is quite large, railway tracks must be divided into several sections and health of each section must be monitored. This health information can then be sent to the nearby railway station in real-time to detect if there is any failure in the railway tracks.

II. Technical Background

The relaying of the health information to the railway station can be done using wired or wireless communication. Wired communication is expensive and to pinpoint the fault location is not possible. To detect the railway track breakage, we can run a continuity test and can determine if a wire, stuck with the railway track along its length, is broken or not. This gives only the binary information about the railway track i.e. if it is broken or not but not exactly where it is broken. Out of many possible types of faults in the railway tracks, we are focusing on the breakage such that the two parts of the railway tracks are not in contact with each other anymore. Such a fault is shown in Fig. 1.

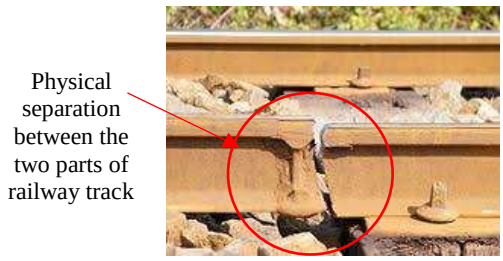


Fig. 1. A broken railway track, there is no physical contact between the broken sections taken from [3].

Another option is to use the wireless communication. For each section, one wireless node is connected. In case of breakage, the packet is initiated by the mote closest to the breakage and the packet is then routed to the nearest railway station. Fig. 2(b) shows an example of such communication. Because of the advancements in the enabling technologies especially in the semiconductor industry, this wireless communication is getting cheaper day by day and hence presents a promising candidate to be used for long tracks for breakage detection. Using the mote address which initiated the packet, the fault can also be located thus efforts and time to find and rectify the fault will be considerably reduced.

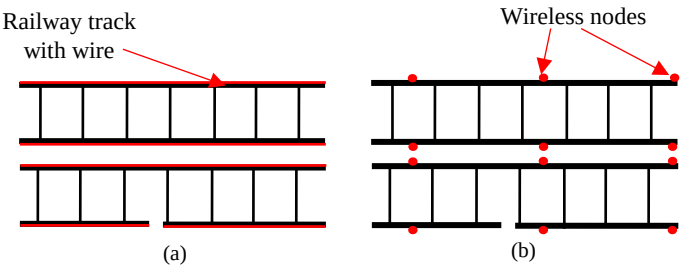


Fig. 2. (a) shows a healthy track with wired connected along the length (in red), the bottom track is broken and hence the wire is also broken which can be detected using a continuity test. (b) shows the same track equipped with sensor nodes (red dots). Using a sensor the breakage in the track can be detected.

III. Objective

The goal of the project is to present a cheap, fast and energy-aware system to detect the arrival of the train as well as to monitor the railway track health using a wireless sensor network. In case of a fault, the packet is routed to the nearby railway station and any catastrophe because of the damaged track can be avoided. The system provides real-time status of the railway track assisting in the timely actions in case of railway track breakage.

IV. Train Arrival Detection and Breakage Detection

The idea behind the detection of train arrival and detection of breakage faults is based on the fact that a moving train induces vibrations in the tracks. These vibrations can be easily detected and can serve as a robust signal to determine if a train is coming or not. To detect these vibrations, a cheap vibration sensor which usually consists of a piezoelectric material can be used. Because of the length of the railway track and the precarious nature of the application, the wireless motes must be cheap, robust, small in size, and must have low-power consumption. Since no intensive calculations are to be made, computing performance of the motes is not critical.

As said earlier, the transfer of the health information is done using wireless communication. The railway track is divided into several sections and one mote is placed on each section such that the part of the railway track between two consecutive motes say mote 2 and mote 4 is monitored by these motes as shown in Fig. 2(c). All the motes construct a wireless sensor network.

The detection of the vibrations by consecutive motes show that the railway section between those motes is not broken. However if one of the motes detect the vibrations but its next mote does not, there is a breakage between those motes. In that case, a fault must be reported to the station as soon as possible.

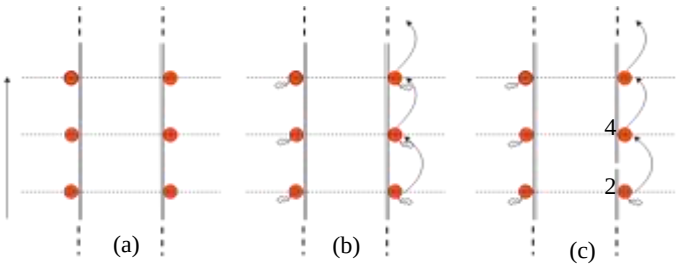


Fig. 3(a) shows the placement of the motes (red circles) on both sides of the railway track. (b) shows how the packet is routed using multiple hops to the sink mote when vibrations are detected on the track. In case of no fault, all the motes sense the vibrations. (c) The railway track is broken between motes 2 and 4, thus mote 2 will detect the vibrations and the other motes on this side of the track will not.

V. Architecture

All the motes in the network are similar except the gateway mote which will be placed in the railway station control room connected to the screen. All the motes except the gateway mote are called as 'field motes' as they are placed on the railway track in the field. The motes are placed on both sides of the track as already shown in Fig. 3. This is essential to monitor both sides of the railway track.

The sensor motes detect vibrations using connected vibration sensor and send the value (0 or 1) to the gateway mote. Gateway mote will then do the calculation on this data and based on the comparison of the vibration values of two consecutive motes, it decides that whether that part of the railway track is damaged or not. The gateway mote is connected to a PC using a serial cable and the output will be displayed graphically on Graphical User Interface (GUI).

A. *Hardware*

1) Re-Mote Platform

Zolertia™ Re-Mote wireless modules are used in this project. The RE-Mote is a platform popular in both the academy and industry in sensor network applications. Fig. 4 shows the Re-Mote platform. Their features include small size, low power consumption, low cost and easy programmability.

2) Vibration Sensor

To detect the arrival of the train, we need to detect the vibrations in the railway track. For that, a vibration sensor available from Zolertia is used. This vibration sensor contains a piezoelectric transducer. The displacement of the transducer from the mechanical neutral axis results in the bending which in turn induces strain within the piezoelectric element and generates voltages. This is an analog vibration sensor which connects to the analog input of the mote, an ADC then converts the analog voltage signal to the digital voltage signal [5]. The used vibration sensor is shown in Fig. 5.

Fig. 4. Zolertia Re-Mote Platform taken from [4]

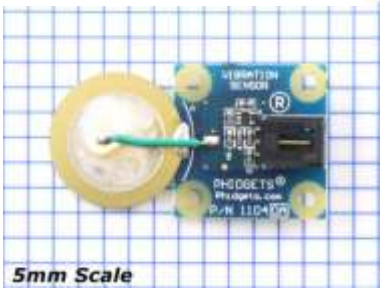


Fig. 5. An image of the used vibration sensor. The circular part detects the vibration, adapted from [5].

B. *Classification of the System*

The proposed system can be classified into the following categories:

1) Multi-hop system: Since the distance between the field motes and the gateway mote is too large for single-hop, multiple hops are essential to route the packet to the destination.

2) Manual node deployment: The motes are deployed manually on the tracks and the positions are static. Network discovery phase is still implemented to avoid any dependencies on the mote IDs.

3) Scalable: New motes can be added in the network without any problem since the routing is independent of the mote IDs. Hence the system is scalable. This is important only when the railway track itself is extended.

4) Time-Driven data reporting method: Each mote broadcasts a packet after a certain time for routing decisions. Also, the vibrations are detected periodically. The vibration detection can also be event-driven but in our application, the vibrations are persistent and not intermittent, hence they can be detected at any time such that any decision based on the information received at the gateway is taken within time. Thus the data reporting method is time-driven.

5) Hierarchical network routing: In our system, all the computations are done at the gateway mote hence the system is termed as hierarchical. This allows the use of cheap field motes with minimum computing power.

VI. MAC Layer Configuration

In this application, it is important that the packet must reach the gateway under all conditions. It is also important to minimize the power consumption. Hence retransmissions must be minimized and for that, Carrier Sense Multiple Access (CSMA) is used as the MAC layer protocol which proactively senses the medium for any other transmission before sending the packet and is more robust than simple ALOHA [6]. ContikiMAC is used as the RDC protocol to further save power. It makes sure that the packet is retransmitted in case of collision and the duty cycle is adapted to save power [7].

VII. Routing

Routing is the method of finding a path over which the packet will travel to reach the destination. The routing protocol is the protocol through which the motes exchange the information required to find the best route. The method to calculate this path is termed as the routing algorithm.

A. Network Topology

The system comprises of sensor motes equipped with external batteries and vibration sensors. These sensor motes are connected wirelessly and communicate with each other on a particular frequency. The system is built using partial mesh network topology that is each mote is connected to all its neighboring motes that are in its transmission range. This topology provides redundancy and allows a greater physical network spread.

B. Routing Protocol

The main target is to send data from each individual mote to the gateway mote using network routing protocol where data will be displayed in user understandable form using a GUI. Each mote will send data to its neighbor with the least cost and that mote will then forward this packet to the other motes based on the routing protocol and eventually it will reach the gateway.

The routing protocol is implemented in the following way. Each mote broadcasts its packet after every 10s. The broadcast timing is randomized to avoid congestion of the system thus fewer collisions are expected. This packet will be received by the neighboring motes in the range of the sender. The broadcast packet contains the sender's ID, the Received Signal Strength Indicator (RSSI), the battery level of the sender, and the cost to route the packet to the gateway. After receiving the broadcast message, the mote calculates the cost of using the mote from which the packet is received as the route. If the calculated cost is smaller than the cost already saved in the routing table, the mote from which the packet is received is declared as the next hop. How this cost is calculated is determined by the routing algorithm discussed next.

C. Routing Algorithm

1) Network Discovery Phase

Initially, no mote has information about its neighboring motes and will go through the network discovery phase. Each mote has a lookup table containing the next hop address, the total cost of the path to gateway mote and its own battery level. The destination address which is the address of the gateway mote is fixed and is known to each mote beforehand. Hence the aim of each mote is to send its packet to this fixed gateway mote using the path with the least cost.

This lookup table is initialized with next hop as any arbitrary address, very high cost (ideally infinite) and battery value as 100%. For gateway mote, the cost is fixed to 0. As described, each mote broadcasts its routing table every 10 seconds in order to inform its neighbors about its presence. Now each mote has to update its lookup table based on the best available path-cost to sink mote. The general formula for calculating the total cost to sink mote is given in (1)

$$Total\ Cost = Local\ Cost + Received\ Cost \quad (1).$$

Received cost is the cost of the path between sending mote and sink mote and is received in the broadcast packet. Local cost is the cost between sending mote and receiving mote and is calculated using (2)

$$Local\ Cost = (Max\ RSSI - Received\ RSSI) + (Max\ Battery - Received\ Battery) \quad (2).$$

So total cost is the overall cost from the current mote to the sink mote if the sending mote is chosen as the next hop. The lower cost indicates the better path. Fig. 6 shows the routing protocol in which the path with the best battery and highest signal strength is chosen. The routing table starts updating gradually from the motes closer to gateway towards the far

motes. Let us start with the motes that are in the transmission range of sink mote. When these motes receive a broadcast packet from the gateway, they will calculate the cost to the gateway and update the routing table with next hop as the gateway and the cost to be broadcasted later on. The cost to the sink mote for these motes will, in any case, be less than the cost to send the packet via other motes since in the former, only a single hop is required. Now, these motes will broadcast their routing table to other motes which are in their transmission range. The receiving motes will update their routing tables based on the best cost and the process continues until all the motes update their tables. The flow of the process on receiving a broadcast message is shown in Fig. 7.

Upon detection of the vibration, a mote will unicast a packet to the next hop according to its routing table. Next hop then forwards the packet to its next hop until the message reaches sink mote where it will be stored in an array. Fig. 9 shows the algorithm to send the unicast packet.

VIII. Gateway Functionality

The computations for the detection of the train and detection of the faults is done by the gateway mote. The gateway mote receives the broadcast message from all the motes in its range. These broadcast messages are not important and do not play any role in determining the train arrival or fault detection. The unicast messages received by the gateway (which were initiated by the mote which detected the vibrations) are important. A unicast message consists of the address of the mote which initialized the packet. This is all the information gateway needs to determine the train arrival or for fault detection.

After a certain time, gateway runs an algorithm to consume the information received in unicast messages. A Boolean array of size n for n number of motes saves 1 in its mth position when a unicast packet is received having source ID m. This indicates that the detection of vibrations by the mth mote.

Fig. 8 shows the example when the train is arriving and the track has no breakage. In that case, all the motes will sense the vibrations and the gateway will receive a unicast packet from all the motes. Hence each position of the array will store $true$. To detect the faults is also simple. We just need to find two different values at consecutive positions of the array. The location of the fault is also found out using those positions. Fig. 8 shows such a scenario where the fault exists between mote 2 and 4.

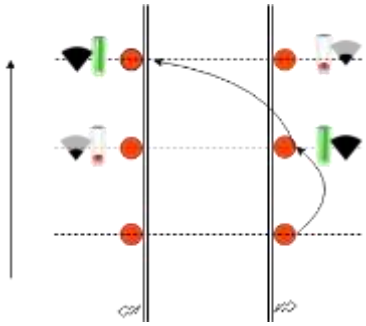


Fig. 6. Routing protocol based on signal strength and battery life. Packet is routed using the path with highest battery and best signal strength.

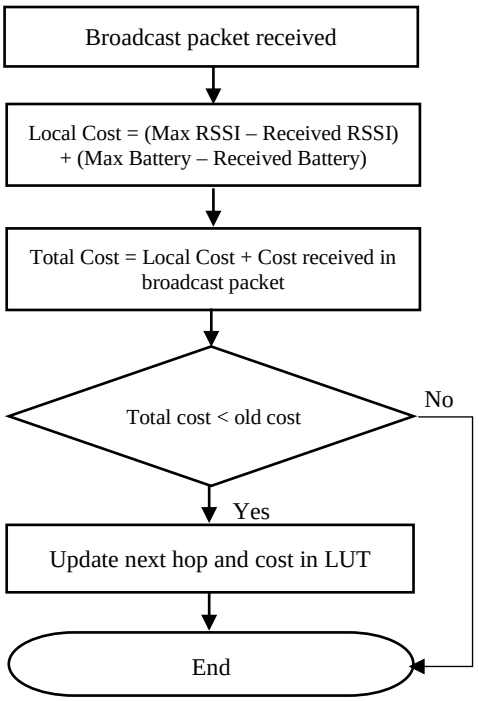
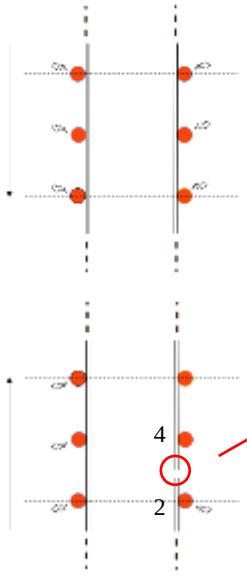


Fig. 7. The algorithm that runs when a broadcast packet is received is shown. The value of total cost is calculated which determines if the next hop in the LUT is to be updated or not.



Mote ID	Vibrations Detected
1	True
2	True
3	True
4	True
5	True
6	True

Mote ID	Vibrations Detected
1	True
2	True
3	True
4	False
5	True
6	False

Fig. 8. The top figure shows the arrival of the train. All the motes are vibrating and they unicast their packet to the gateway. The gateway accordingly updates its array. All the entries show true which can be construed as the train arrival. Bottom figure shows the case in which a breakage exists between motes 2 and 4. No vibrations are sensed by mote 4 and hence no unicast packet is received on gateway from mote 4. Thus the corresponding entry in the table shows false. 2 different values in corresponding motes indicate the fault. The location of the fault can be determine easily using the table entries.

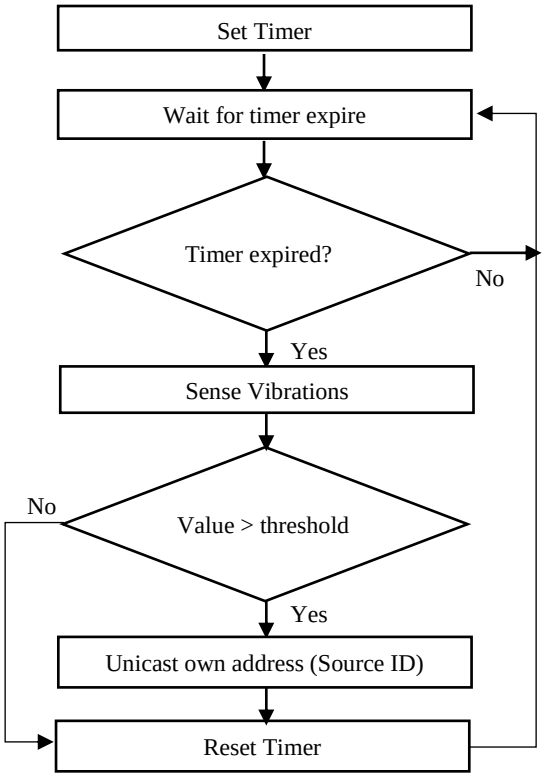


Fig. 9. The periodic algorithm to sense the vibrations and initialize the unicast packet is shown. The only information to be sent to the gateway is the source ID which initiaes the packet.

IX. Additional Features

A. *Fault Detection*

For robustness, the system should be able to detect the mote failures such that the routing path could be changed accordingly. There are multiple ways to detect when a mote fails. In our system, this is done in the network layer.

A simple implementation is to re-initialize the routing table of each mote periodically with the highest cost for each mote. It is done to avoid the situation when the failed mote was the next hop in some other mote i.e. the failed mote had the least cost. If failure recovery is not implemented, the mote will keep sending the packet to the failed mote (since it has the least cost) and that packet will be lost unless and until another mote provides a better cost to replace the failed mote in the routing table. Hence the routing table periodic re-initialization is done to handle this situation proactively. After a certain time, the whole table is re-initialized with a high cost which then gets updated according to the upcoming broadcast messages. The neighboring motes will not receive broadcast messages from the failed mote and that failed mote can never be set as the next hop. Instead, the mote which *now* provides the least cost is set as the next hop.

Another method which could be implemented is to wait for certain numbers of broadcast received packets. If in those packets, say mote A's packet does not appear it means that mote A is faulty. This can be implemented using a counter on the receiving mote for every other mote. Whenever a broadcast packet is received, the sender resets the counter of the sending mote and increments the counter of all other motes. When the counter of any mote reaches a maximum say 20, it shows that for last 20 broadcast, no packet is received from that mote and hence it must be faulty or temporarily out of the range. Thus the cost of only this mote can be set to very high such that some other mote with lower cost can replace it. Another advantage of this implementation is, we can have extra information about the incoming broadcast traffic on each mote.

B. *Energy-Awareness*

Since battery level is taken as a parameter to calculate the cost, the routing is battery-aware. This is important to extend the lifetime of the system.

C. *Dynamic*

The routing decisions are based on the signal strength and battery level of the motes hence a packet can take different paths to the gateway mote by examining which path has the lowest cost. Also in the case of a fault, the faulty mote is eliminated from the network and a new path should be formed to route the packet.

D. *Scalability or Network Expansion*

Although manually deployment is done, the system is self-organized and a new mote can be easily added to the network. This is possible because the protocol is independent of the mote's IDs. Once a new mote gets active, it will start receiving routing information from the neighboring motes in transmission range. Based on the best cost, it will update the routing table.

X. Summary

A simple wireless sensor network to detect the arrival of the train as well as to detect the presence of breakage in the railway tracks is discussed. An algorithm to determine the train arrival and fault detection runs periodically on the gateway mote and the array in the gateway mote is re-initialized after the algorithm. Thus the health of the railway tracks is periodically updated in real-time and is displayed by the gateway. Real-time operation facilitates the timely decisions to avoid any accident due to the breakage. Re-initializing the routing table periodically provides the elimination of the faulty motes from the network. This method provides an easy way to locate the fault hence the time for maintenance is reduced.

References

[1] Esveld C., "Modern Railway Track", MRT Productions, 2001.

[2] Wirtu, L Bayissa, Dhanasekar M., "High speed detection of broken rails, rail cracks and surface faults", CRC for Rail Innovation

[3] Wikipedia – the free encyclopedia, "Rail Inspection", https://en.wikipedia.org/wiki/Rail_inspection, online accessed 04.07.18

[4] Zolertia Re-Mote Platform, 03.09.16, https://github.com/Zolertia/Resources/wiki/RE-Mote, online accessed 11.07.18

[5] Z1 Sensors, http://zolertia.sourceforge.net/wiki/index.php/Z1_Sensors, online accessed 11.07.18

[6] Bachir, A, et al. "MAC essentials for wireless sensor networks." IEEE Communications Surveys & Tutorials 12.2, 2010.

[7] Dunkels, A. "The contikimac radio duty cycling protocol.", 2011